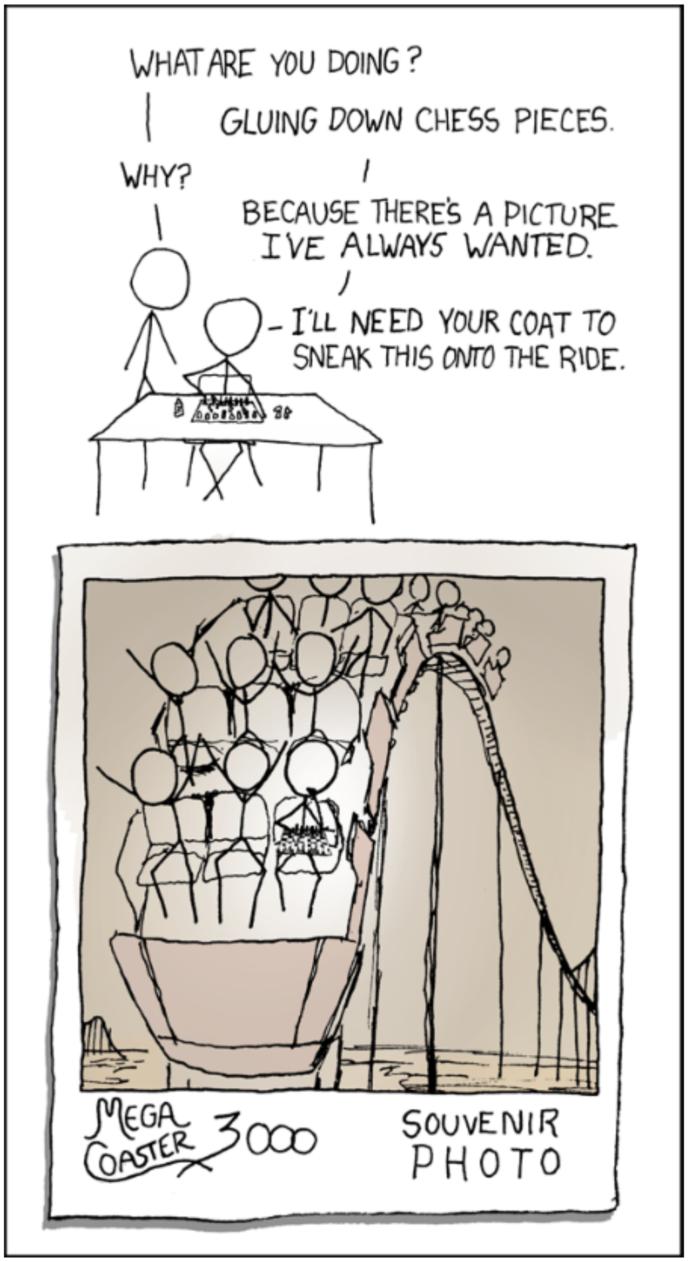
HiMCM 2018: Problem A

Team 8604

November 16, 2018


WHAT ARE YOU DOING?
GLUING DOWN CHESS PIECES.
WHY?
BECAUSE THERE'S A PICTURE I'VE ALWAYS WANTED.
I'LL NEED YOUR COAT TO SNEAK THIS ONTO THE RIDE.
MEGA COASTER 3000
SOUVENIR PHOTO

https : // xkcd.com / 249 /

Table of Contents

Summary

There are many roller coaster rating/ranking sites online that, while taking some objective measures into account, heavily rely on subjective input to determine the rating or ranking of a particular roller coaster (e.g., an “excitement” or “experience” score of an “expert” rider to measure “thrill”). Though such reviews take into account multiple aspects of a roller coaster and how they work together to provide an experience, these reviews often include subjective opinions on roller coasters rather than their actual performance.

To address this problem, we developed an algorithm which could rank roller coasters solely based on objective measures such as height or length. Our algorithm includes a four-dimensional vector with each dimension representing scaled height, speed, length and duration. The length of this vector is calculated and scaled to determine a base score. Additionally to the base score are “bonus” scores, which are scaled and weighted values relating to g-force, number of inversions, vertical angle, and drop height. These values, positive or negative, are averaged collectively to create one “bonus” score, which is then added to the base score. This final score is our definition of a given roller coaster’s level of thrill.

This algorithm is then used to sort all of the given roller coasters in descending order, and the first 10 coasters are selected as they are the 10 most thrilling. Finally, we designed a mobile application which implements this algorithm to help the user find a roller coaster to go ride based on the user’s preference of travel time and thrill.

RollerHoster Application

After launching the app for the first time, the landing page is a profile set-up form. The function of a profile is to save how far the user is willing to drive and what range of thrill they are willing to experience in a given coaster. The range of miles the user is willing to travel and the range of thrill is set by two sliders. The range of thrill is expressed in arbitrary units 1 to 5. To convert from the thrill units used by the application to the units used by our algorithm, the following function is used: $f(x) = (\max - \min)\left(\frac{x-1}{4}\right) + \min$.The division $\frac{x-1}{4}$ transforms the input into a percent, and the coefficient and y intercept (max-min) and + min transform the percent into the units used by the algorithm. An additional function of the profile is to write reviews. This will be discussed in more depth later.

Once profile creation has been finished, the user will remain logged in until the app is uninstalled. The new landing page is a Google Maps map or similar map with pins placed on the map with labels representing a roller coaster within the qualifying distance and thrill intervals specified by the user. Clicking on a pin will provide more detail about the roller coaster and associated park through a pop-up screen. Such information includes the name of the ride, name of the park where the coaster is located, street address of the park, name of the park, construction and type of the coaster, physical information about the coaster, reviews of the coaster, pictures of the ride and park, line wait times, cost for parking and entrance to the park, and our calculated thrill score on the scale 1 to 5, calculated with the function $g(x) = 4 * \left(\frac{x-\min}{\max-\min}\right) + 1$, such that $f(g(x)) = x$. Physical information includes height, length, speed, number of inversions, biggest drop, duration of ride, maximum g force reached by a rider during a typical ride, and largest vertical angle reached during a typical ride. Line wait times will be expressed as the expected time one would have to wait in line before they could ride the coaster, the average wait time, and a box to textually input a day of the week and time and return a predicted wait time. The data to calculate these wait times would need to be gathered for the construction of this app.

The review system allows for users with accounts to place star reviews out of 5 as well as a textual response. This allows for the user to base their decision off of attending a given roller coaster subjectively from a centralized source of reviews. Users are able to like and dislike reviews of coasters, and reviews will be sorted by this like ratio, so that future users will have access to helpful reviews as easily as possible.

The final tab of the app is a list of potential roller coasters, sorted. This list is sorted in order of $y = \frac{\text{thrill of } y}{\text{travel time to } y}$. This ratio maximizes thrill while minimizing travel time. This is necessary because it is a waste of time to travel very far to go to an especially thrilling coaster, as well has traveling a very short distance to a terrible roller coaster. The user can click on individual coasters in the list to access more information, this feature being described in the paragraph above. An additional button for each item of the list will open the Google Maps app and provide directions to this roller coaster from the user’s current location.

** The described UI can be seen in Appendix C.

News Release

New Roller Coaster App Helps Users Find the Best Rides

Born from the creative minds of the Team 8604 HiMCM contestants, the new app RollerHoster helps users find the perfect roller coaster. The app uses one’s preferences to find the best roller coaster for him/her. The app asks how far one is willing to drive and how much thrill one wants to experience, and plugs the data into a powerful algorithm designed to find roller coasters that match your preferences as close as possible. First, it finds all roller coasters that meet the user’s specifications. It then sorts the potential coasters to maximize thrill for one’s travels. But how can one define how thrilling a roller coaster is? RollerHoster has solved this too. The algorithm used by the app takes into account many factors such as the duration of the ride, the number of inversions, and the maximum speed experienced to calculate a thrill-value. This thrill-value is divided by the travel time to make a ratio that can be compared with that of other coasters.

If the user if interested in one of the suggested roller coasters found on either the local map view or the list format, he/she can view a comprehensive list of specifications about the ride and associated park. Users can also view a full list of rankings of all the coasters in RollerHoster's database. This allows the user to compare between many coasters and manually pick which one they would prefer.

But which roller coasters are the most thrilling? RollerHoster can tell you. The best roller coaster in the world is Steel Dragon 2000, located in the Nagashima Spa Land park. This monster coaster is the longest in the world at over 8000 feet long, and its top speed is almost 100 mph. Steel Dragon 2000 has a rating of a 5 on the RollerHoster app. The second best coaster is Kingda Ka. 28 seconds of pure speed and drop is an accurate description of this crazy coaster. Kingda Ka has a rating of 4.7 on the app.

RollerHoster is the best way for anyone to find a ride that will give them a thrill while also making sure that the ride contains all the elements the user desires in a ride. The app will completely change and enhance the theme park experience for everyone as soon as it is released.

Introduction

Many roller coaster rating/ranking sites online rely heavily on expert opinion to rank roller coasters. This often results in the subjective ranking of roller coasters. Here, we rank roller coasters solely based on the physical attributes of roller coasters to eliminate subjectiveness and to provide an accurate ranking of roller coasters. To do this, we had to make the following assumptions:

Assumptions

Null values are factored out of the algorithm rather than set to zero

Some pieces of information were not provided from the excel file and could not be found from databases online (https://www.rcdb.com/, https://www.ultimaterollercoaster.com/, and https://coasterpedia.net/). In such cases, the null value was factored out the the equation and the ranking of the roller coaster was based solely on the pieces of information given.

Location of the roller coaster does not affect experience

Roller coaster preferences are solely based on the enjoyment of the ride rather than the environment surrounding the ride.

Material of roller coaster construction will not influence experience

There is no significant difference between wooden and metal roller coasters, so they will not be factored into the algorithm.

The roller coaster type will not affect base algorithm

The base algorithm will not be influenced by whether the rInoller coaster is a stand up roller coaster, sit down roller coaster, or inversion roller coaster because these values are not continuous.

A roller coaster is assumed to be equally exciting through the whole ride

Assuming the roller coaster is able to thrill riders throughout the whole ride, the algorithm assumed that the longer the ride was, the more thrilling it was.

■ Definitions

Thrill

The degree of thrill from experiencing a given roller coaster ride.

Null

An empty data point where data is not given.

Speed

The maximum speed of a roller coaster reached during the ride. Speed is measured in miles per hour.

Height

The maximum height reached by the roller coaster during a ride. Height is measured in feet from ground level.

Length

The length of the roller coaster ride. Length is measured in feet.

Inversions

The number of times the track turns the rider upside-down and back during a ride.

g-force *(F_g)*

The maximum g-force reached within the roller coaster ride, where g-force is the measure of acceleration at a given time.

Duration

The total duration of a roller coaster ride measured in seconds.

Vertical Angle

The largest angle below the horizon the track makes.

Methodology

$$\text{thrill} = 100 * \left(\sqrt{\frac{S_s{}^2 + L_s{}^2 + H_s{}^2 + t_s{}^2}{4 - \text{null}_1}} \right) +$$

$$100 * \left(\frac{1}{4 - \text{null}_2} \right) \left(\left(\frac{\vartheta - \vartheta_{\text{avg}}}{\vartheta_{\text{max}} - \vartheta_{\text{avg}}} \right) \left(\frac{100 * \text{tot}_{\vartheta}}{\text{tot}_{\text{coast}}} \right) + \left(\frac{\text{inv}}{\text{inv}_{\text{max}}} \right) \left(\frac{100 * \text{tot}_{\text{inv}>0}}{\text{tot}_{\text{coast}}} \right) \right.$$

$$\left. + \left(\frac{F_g - F_{g-\text{avg}}}{F_{g-\text{max}} - F_{g-\text{avg}}} \right) \left(\frac{100 * \text{tot}_{\text{Fg}}}{\text{tot}_{\text{coast}}} \right) + \left(\frac{d - d_{\text{avg}}}{d_{\text{max}} - d_{\text{avg}}} \right) \left(\frac{100 * \text{tot}_{d}}{\text{tot}_{\text{coast}}} \right) \right)$$

To rank the roller coasters, we first developed an algorithm which would take all of the given data into account when ranking the roller coasters, but not make any assumptions about any pieces of data which were not given. The first term of this algorithm is the length of a four dimensional vector with the scaled dimensions speed, length, height, and time. These dimensions were chosen because they are all continuous variables that are independent of track shape and direction. The variables used in this expression are defined as follows.

$S_s = \frac{\text{Max Speed of Given Coaster}}{\text{Max Speed in Given Data Set}}$, $L_s = \frac{\text{Length of Given Coaster}}{\text{Max Length in Given Data Set}}$, $H_s = \frac{\text{Max Height of Given Coaster}}{\text{Max Height in Given Data Set}}$, $t_s = \frac{\text{Duration of Ride on Given Coaster}}{\text{Longest Ride in Seconds in Given Data Set}}$

This vector will have less dimensions if data is omitted from the given roller coaster. The distance is calculated using the formula $\sqrt{S_s{}^2 + L_s{}^2 + H_s{}^2 + t_s{}^2}$ The denominator of the expression, 4 - null$_1$, scales the distance to a value between 0 and 1. 4 - null$_1$ means four original dimensions minus the number of null dimensions. This way, the value will also be between 0 and 1 even if only 1 dimension is given. This value is then multiplied by 100 for ease of use. Our final calculated thrill score will be between 0 and 100, plus or minus “bonus” points.

The remaining terms are what we considered “bonuses”. We had two methods of calculating “bonus”: “aggressive” and “positive”. The aggressive method gave the roller coaster additional bonus points in our algorithm if it had above average data, and removed points if the data was below average. The positive method rewarded any non-zero value. The bonuses for vertical angle, force of gravity, duration, and drop are calculated with the aggressive method, because we believe that having above-average values for these parameters is part of what makes an above-average roller coaster. Thus, above average values should provide additional points to an above-average roller coaster. Conversely, a below average roller coaster will have below average values, thus its level of thrill should be docked points. The bonus points for these terms are calculated with $\left(\frac{x - x_{avg}}{x_{max} - x_{avg}}\right)\left(\frac{tot_x}{tot_{coast}}\right)$ The term in the left set of parentheses is a fraction representing the given roller coaster's value's distance from the overall average value, over the largest distance from the average value overall. This fraction is then multiplied by another fraction representing how relevant the type of value is. There are many null values in the given data, so we decided to relate the amount of non-null values of type x to the total amount of roller coasters. This fraction gives us a method of weighting each type of variable. The bonus points from the number of inversions are calculated using the positive method. This is because we do not believe inversions are necessary to making a thrilling roller coaster. The formula for the inversion bonus points is $\left(\frac{inv}{inv_{max}}\right)\left(\frac{tot_{inv>0}}{tot_{coast}}\right)$The fraction in left set of parentheses is the percentile of the given roller coaster's number of inversions. The right term is the relevance of inversions, this time based on how many roller coasters have more than zero inversions over the total number of roller coasters.

All calculated bonus points are then averaged, multiplied by 100 for ease of use, and added to the other term of the equation. This final sum is the quantified thrill of the given roller coaster.

Where a lone variable (such as ϑ or inv) stands for that factor of a given roller coaster, tot_{coast} stands for the total amount of roller coasters, and $tot_{subscript}$ represented the number of roller coasters which provided the data of the subscript. The other variables were defined as:

Criteria	Max	Avg
Speed ($\mathbf{S}$)	Maximum speed of given coasters in mph	Average speed of given coasters in mph
Height ($\mathbf{H}$)	Maximum height of given coasters in feet	Average height of given coasters in feet
Length ($\mathbf{L}$)	Maximum length of given coasters in feet	Average length of given coasters in feet
Inversions (inv)	Maximum number of inversions by given coasters	Average number of inversions by given coasters
g – force (F_g)	Maximum g – force of given coasters	Average g – force of given coasters
Duration (t)	Maximum duration of given coasters in seconds	Average duration of given coasters in seconds
Vertical Angle (ϑ)	Maximum vertical angle of given coasters in degrees	Average vertical angle of given coasters in degrees

Drop Height (d)	Maximum drop height of given coasters in feet	Average drop height of given coasters in feet

Results

Top 10 from algorithm

Rank	Roller Coaster Name
1	Steel Dragon 2000
2	Kingda Ka
3	Fury 325
4	Top Thrill Dragster
5	Formula Rossa
6	Leviathan
7	Millennium Force
8	Soaring Dragon & Dancing Phoenix
9	Intimidator 305
10	Coaster Through the Clouds

Top 10 from website 1, USA only (https://www.thetoptens.com/best-roller-coasters/)

Rank	Name of Roller Coaster	Park
1	Millennium Force	Cedar Point
2	Top Thrill Dragster	Cedar Point
3	Superman the Ride	Six Flags New England
4	Beast	Kings Island
5	Oblivion	Alton Towers
6	Superman : Escape from Krypton	Six Flags Magic Mountain
7	Drachen Fire	Busch Gardens
8	Goliath	Six Flags Magic Mountain
9	Son of Beast	Kings Island
10	Superman – Ride of Steel	Six Flags Darien Lake

In comparison with the top ten list generated by our algorithm,the first outside list,from the University of Buffalo, shares two of the same coasters. This list is very different from our list as it has eight coasters not included in our listand three coasters not in our given data set. This difference can be attributed to the website only including roller coasters of USA while our list determined the top 10 roller coasters world wide. The number one rank in the University of Buffalo list is Millennium Force.

This coaster was included in our list as well,but it was placed in the seventh rank. The number two rank in the University of Buffalo list was awarded to Top Thrill Dragster,which was placed at fourth in our list. This list helps to demonstrate the difference in rankings that can come through subjective reviews,whereas our list is purely objective.

Top 10 from website 2, USA and Canada only (http://www.eng.buffalo.edu/~hulme/topcoast.html)

Rank	Name of Roller Coaster	Park
1	Millennium Force	Cedar Point
2	El Toro	Six Flags Great Adventure
3	Intimidator 305	Kings Dominion
4	Fury 325	Carowinds
5	Maverick	Cedar Point
6	Voyage	Holiday World
7	Kingda Ka	Six Flags Great Adventure
8	Skyrush	Hersheypark
9	Leviathan	Canada ' s Wonderland
10	Superman the Ride	Six Flags New England

The second top ten list we looked at was more comparable to the list generated by our algorithm. This list, from The Top Tens, shares three (Kingda Ka, Fury 325, and Millennium Force) of the same coasters with our list. This list also contains coasters that are all in our given data set, but the top 10 list of website two only includes USA and Canada roller coasters, leading to some inevitable differences between our list and website 2’s list. The number one rank in this list was awarded to Millennium Force, similar to the first outside list we looked at. This is may be because our list does not rely on any subjective data and therefore cannot know the experiences of people who have ridden it in the past.

Strengths of our Model

Model makes no assumptions about null data

If incorrect assumptions were made about a roller coaster's data, the algorithm could artificially lower the score. This would lead to many good coasters being ranked lower due to them not providing full data. Making no assumptions and ignoring null data will lead to more just scores for all coasters, even those with missing data.

All continuous variables are factored into the algorithm

Factoring all continuous variables into the algorithm allows it to be most representative of all coaster types.

All bonus points are weighted by relevance

By weighting all the bonus categories by relevance, data that is not represented in many coasters is considered less valuable, allowing for more equal scoring between the coasters.

Algorithm requires little computation power

The algorithm takes very little power to compute score for a given roller coaster. This allows for the algorithm to be run on a variety mobile devices, making the application more versatile and accessible.

Scores can be directly compared between separate roller coasters

Scores are on a linear scale between 0 - 100, allowing for direct comparison between any set of coasters. This benefits the application because coasters can be easily organized and ranked based on their scores.

Limitations of our Model

Missing data is completely ignored

Because the algorithm developed only ranked roller coasters based on given data, roller coasters with missing data will be ranked solely on the data that is provided, meaning that missing data will be completely ignored. In the case that the missing data significantly changes a roller coaster's rank, the algorithm will not be able to accurately rank the roller coaster. Our alternate solution tries to address this problem by predicting missing data based on how the other data points compare to other roller coasters.

Relevance may not be the best weight

Given partially complete data, it is important to determine how much each piece of data will impact the overall ranking of a roller coaster. Because height, length, and speed were given for all of the roller coasters, they were all weighted equally, but factors such as vertical angle, number of inversions, maximum g-force, duration, and drop height were given a weight multiplied based on relevance, which is appropriate because if a low number of roller coasters have the specific piece of data included, it would be unfair to give them a large number of points. Relevance was calculated by the number of roller coasters with that data factor included divided by the total number of roller coasters. The problem with relevance weights is that they base how much the score is worth rather than how important the score is, which could potentially cause errors in ranking.

Descriptive specification data (construction, location, age, design, etc.) was not taken into account

The algorithm ranked the roller coasters only based on quantitative data, so factors such as such as location, park, construction, age, or design of the roller track were no taken into account when calculating which roller coaster was the most thrilling. However, in the app which we have developed, this information including pictures of the roller coaster will be available to for the user to see to better determine which roller coaster would be suitable for the user.

Conclusion

After having reviewed the ranking of roller coasters to ensure the algorithm worked as supposed to, it was decided that the algorithm produced an accurate ranking of roller coasters given that not all pieces of data were given.

As for the application developed to help enthusiasts find and compare roller coasters, the application successfully addresses all required criteria well and allows for the user to effectively search for the most fitting roller coaster as based on their needs.

If there were more time, future extensions could include developing the algorithm further to quantify factors such as steel or wood construction, location, the park that owns the ride, the age of roller coaster and the design of the track. One possible way to measure these qualitative qualities is to base part of the overall score of a roller coaster on reviews from those who have been there. This way, the algorithm takes more of a holistic approach to rank roller coasters.

Appendix A

Top 10 Roller Coasters Physical Data Table

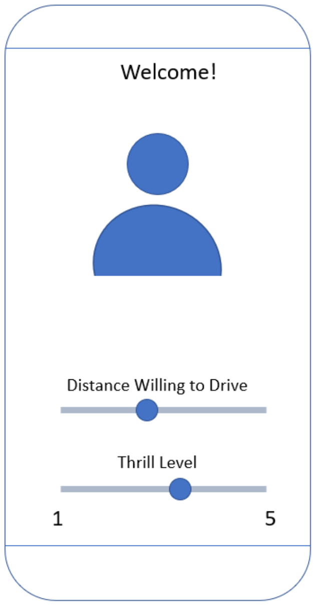
Rank	Roller Coaster Name	Score	Height (ft)	Speed (mph)	Length (ft)	⌗ Inversions	Drop height (ft)	Time (m : s)	G – Force	Vertical Angle (Deg.)
1	Steel Dragon 2000	91.711	318.3	95.0	8133.2	0.0	306.8	4 : 00	Null	Null
2	Kingda Ka	87.766	456.0	128.0	3118.0	0.0	418.0	0 : 28	Null	90.0
3	Fury 325	83.644	325.0	95.0	6602.0	0.0	320.0	Null	Null	81.0
4	Top Thrill Dragster	81.638	420.0	120.0	2800.0	0.0	400.0	0 : 30	Null	90.0
5	Formula Rossa	77.263	170.6	149.1	6561.7	0.0	Null	Null	Null	Null
6	Leviathan	75.298	306.0	92.0	5486.0	0.0	306.0	3 : 28	Null	80.0
7	Millennium Force	79.835	310.0	93.0	6596.0	0.0	300.0	2 : 20	Null	80.0
8	Soaring Dragon & Dancing Phoenix	74.64	196.8	84.0	4192.0	6.0	Null	Null	Null	Null
9	Intimidator 305	72.403	305.0	90.0	5100.0	0.0	300.0	3 : 00	Null	85.0
10	Coaster Through the Clouds	72.359	242.8	84.5	5105.0	0.0	255.9	4 : 12	Null	Null

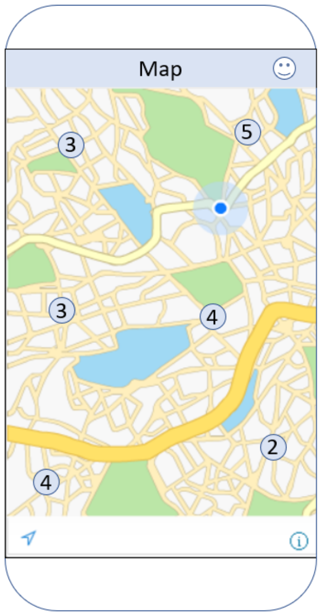
Appendix B

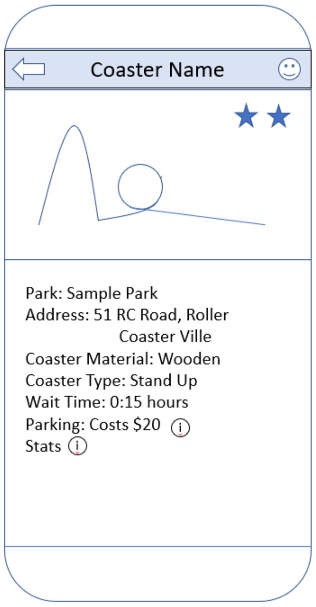
Code (written in Python 3.6.7)

```
  import pandas as pd
from math import sqrt, isnan


class Thrill:
    “””
    Holds a list of each coaster’s scores
    “””

    def __init__(self, length):
        self.length = length
        self.coasters = [0 for _ in range(length)]
        self.points = [0 for _ in range(length)]

    def add(self, values):
        “””
        Adds a list of scores to the existing list of scores. Skips null values
        :param values: 300 length list of scores
        “””
        for i in range(len(values)):
            value = values[i]
            if not isnan(value):
                self.coasters[i] += value
                self.points[i] += 1

    def mult(self, m):
        “””
        Multiplies corresponding elements of the given list and lsit of scores
        :param m: list of numbers to multiply with self.coasters
        “””
        for i in range(self.length):
            if self.points[i] != 0:
                self.coasters[i] *= m[i]
            else:
                self.coasters[i] = 0

    def get(self):
        “””
        Gets list of coaster scores
        :return: list of coaster scores
        “””
        return self.coasters

    def get_coaster(self, i):
        “””
        Gets score for a given coaster, based on index
        :param i:
        :return:
        “””
        return self.coasters[i]


def main():
    length = 300
    thrill = Thrill(length)

    # given columns from csv file
    csv = pd.read_csv(“COMAP_RollerCoasterData_2018.csv”)
    heights = list(csv[“height”][:length])
    speeds = list(csv[“speed”][:length])
    lengths = list(csv[“length”][:length])
    angles = list(csv[“angle”][:length])
    drops = list(csv[“drop”][:length])
    inversions = list(csv[“Number Inversions”][:length])
    time = list(csv[“t”][:length])
    g_force = list(csv[“G Force”][:length])
    names = list(csv[“Name”][:length])

    # convert time to seconds
    for i in range(length):
        t = time[i]
        if ‘:’ in str(t):
            t = t.split(‘:’)
            time[i] = int(t[0]) * 60 + int(t[1])
        else:
            time[i] = float("nan")

    # replace 0s with null values
    for i in range(length):
        inversion = inversions[i]
        if inversion == 0:
            inversions[i] = float("nan")

    def sum_nan(data):
        """
        Sums all values in data
        :param data: list of numbers and probably nulls
        :return: sum of data
        """
        return sum([i for i in data if not isnan(i)])

    def max_nan(data):
        """
        Finds max value of data
        :param data: list of numbers and probably nulls
        :return: max of data
        """
        return max([i for i in data if not isnan(i)])

    def average(data):
        """
        Finda average of data
        :param data: list of numbers and probably nulls
        :return: average of data
        """
        return sum_nan(data) / sum([not isnan(i) for i in data])

    def scale(data):
        """
        Scales all values in data from 0 to 1
        :param data: list of numbers and probably nulls
        :return: scaled list
        """
        m = max_nan(data)
        return [*map(lambda point: point / m, data)]

    def vector_length(v):
        """
        Finds length of vector v
        :param v: a vector
        :return: length of vector v
        """
        nonzeros = sum([i > 0 and not isnan(i) for i in v])
        return sqrt(sum([0 if isnan(i) else i * i for i in v]) / nonzeros)

    def aggressive_bonus(data):
        """
        Calculates the aggressive bonus of given data
        :param data: list of numbers and probably nulls
        :return: aggressive bonus of data
        """
        data = [float(i) for i in data]
        relevance = sum([not isnan(i) for i in data]) / length
        avg = average(data)
        m = max_nan(data)
        return [((i - avg) / (m - avg)) * relevance for i in data]
    def positive_bonus(data):
        """
        Calculates the positive bonus of given data
        :param data: list of numbers and probably nulls
        :return: positive bonus of data
        """
        relevance = sum([not isnan(i) for i in data]) / length
        m = max(data)
        return [((0 if isnan(i) else i) / m) * relevance for i in data]

    # lists of scaled values
    scaled_heights = scale(heights)
    scaled_speeds = scale(speeds)
    scaled_lengths = scale(lengths)
    scaled_time = scale(time)

    # lists of aggressive bonuses
    angle_points = aggressive_bonus(angles)
    inversion_points = positive_bonus(inversions)
    g_force_points = aggressive_bonus(g_force)
    drop_points = aggressive_bonus(drops)
    vectors = list(zip(*[scaled_heights, scaled_speeds, scaled_lengths, scaled_time]))
    vector_lengths = [vector_length(i) for i in vectors]

    # sums and multiplies bonuses and vector length, finds thrill
    thrill.add(angle_points)
    thrill.add(inversion_points)
    thrill.add(g_force_points)
    thrill.add(drop_points)
    thrill.mult([0 if not i else 1 / i for i in thrill.points])
    thrill.add(vector_lengths)
    thrill.mult([100 for _ in range(length)])

    def output_coaster(i):
        """
        Outputs coaster[i] to sys.out
        :param i: index of coaster
        """
        with pd.option_context('display.max_rows', None, 'display.max_columns', None):
            print(csv.iloc[[sorted_indices[i]]][
                ["height", "speed", "length", "Number Inversions", "drop", "t", "G Force",
"angle"]].to_string(
                index=False))

    # sorts all coasters based on algorithm
    sorted_indices = sorted(range(length), key=lambda i: thrill.coasters[i])[::-1]
    top_ten_names = [names[i] for i in sorted_indices[:10]]

    # outputs top 10 coasters
    for i in range(10):
        print(str(i + 1) + ')', top_ten_names[i], "   --->   ", round(thrill.get_coaster(sort-
ed_indices[i]), 3))
        output_coaster(i)
        print()

    print(thrill.get_coaster(80), sorted_indices.index(80)+1)


if __name__ == "__main__":
    main()
```

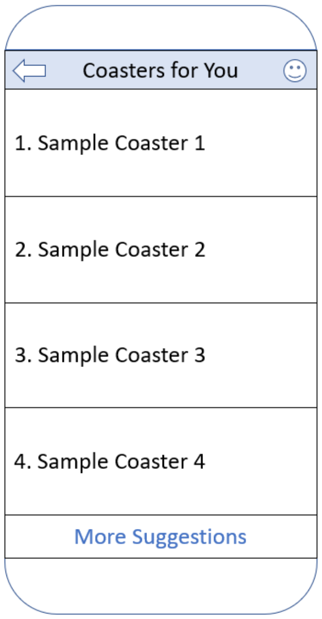
Appendix C







Park: Sample Park
Address: 51 RC Road, Roller Coaster Ville
Coaster Material: Wooden
Coaster Type: Stand Up
Wait Time: 0:15 hours
Parking: Costs $20
Stats



1. Sample Coaster 1
2. Sample Coaster 2
3. Sample Coaster 3
4. Sample Coaster 4

More Suggestions

Appendix D

Works Cited:

Hulme, K. F.Top 10 roller coasters. Retrieved from http://www.eng.buffalo.e-du/~hulme/topcoast.html

Sandoval, N. G.Top ten best roller coasters&nbsp; Retrieved from https://www.-thetoptens.com/best-roller-coasters/